## TrainingData

- m\_trainingDataFile : ifstream
- + TrainingData(filename : const string)
- $+ \sim TrainingData()$
- + isEof(void) : bool
- $+\ getTopology(\&topology: vector{<}unsigned{>}): void$
- $+ \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} + \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} + \hspace{0.1cm} \hspace{0.$
- + getTargetOutputs(&targetOutputVals: vector < double >): unsigned

## Generation

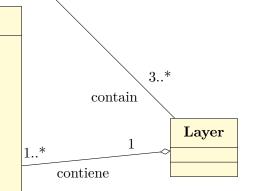
- $m_{training}$ DataFile : ofstream
- + Generation(filename : const string)
- $+ \sim Generation()$

## Network

- m\_layers : vector<Layer>
- m\_error : double
- m recentAverageError : double
- $m_recentAverageSmoothingFactor: double$
- + Network(&topology : const vector<unsigned>)
- $+ \sim Network()$
- + feedForward(&inputVals : vector<double>) : void
- + backProp(&targetVals : vector<double>) : void
- + getResults(&resultVals : vector<double>) : void
- + getRecentAverageError(void) : double

## Neuron

- eta: double
- alpha: double
- activationFunction: double
- activationFunctionDerivative : double
- randomWeight: void
- sumDOW(&nextLayer : const Layer) : double
- m\_outputVal : double
- m\_outputWeights : vector<Connection>
- m\_myIndex : unsigned
- m\_gradient : double
- + Neuron(numOutput : unsigned, myIndex unsigned)
- $+ \sim Neuron()$
- + getOutputVal(void) const : double
- + feedForward(&prevLayer : const Layer) : void
- + calcOutputGradients(targetVal : double) : void
- + calcHiddenGradients(&nextLayer : const Layer) : void
- + updateInputWeights(&nextLayer : Layer) : void



1

holds Connection

1..\*

- weight : double
- deltaWeight : double